



Architektur von Videospiele

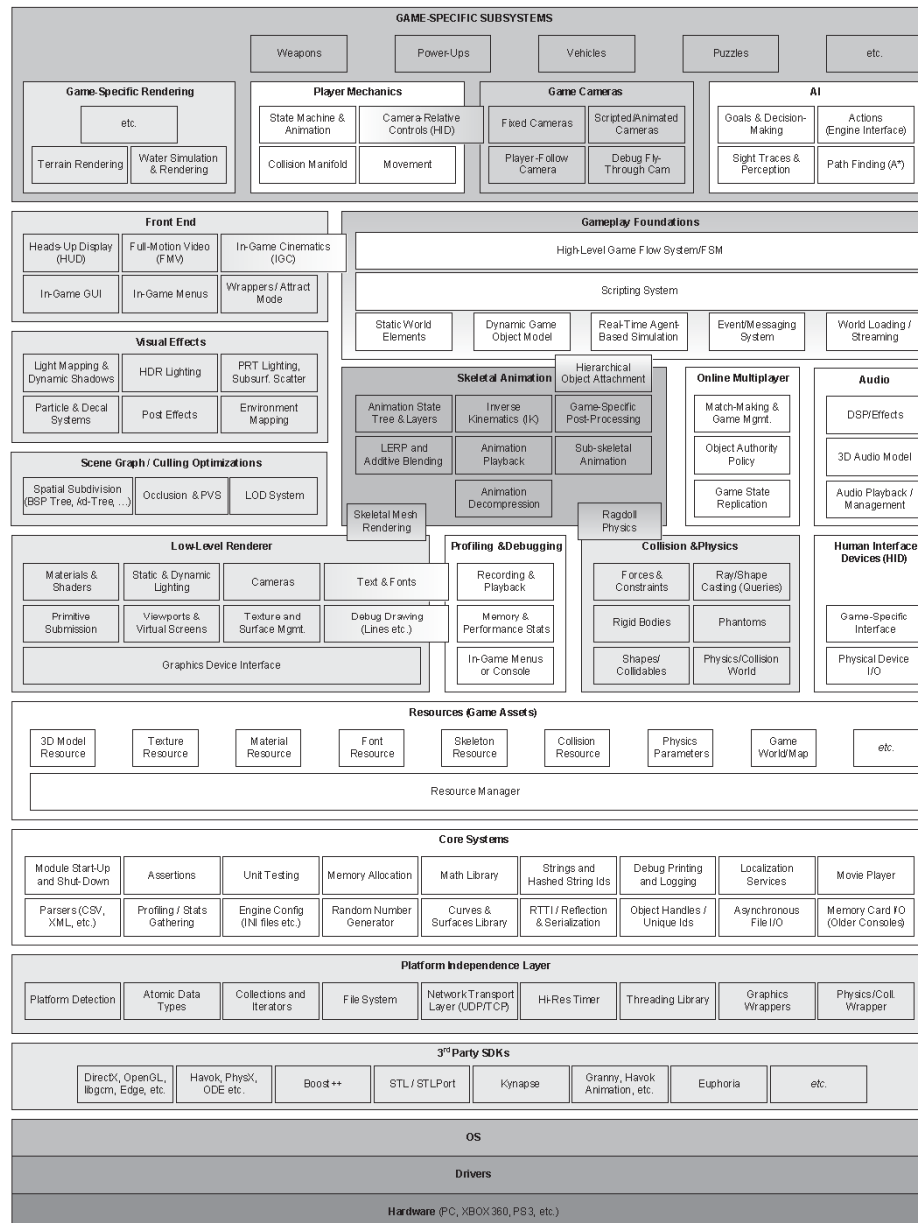


ORGANISATORISCHES & DISCLAIMER



Viele Fragen

- Um was muss ich mich alles kümmern?
 - Während das Spiel läuft (**Runtime**):
Eingabe/Ausgabe, Menüs, Einstellungen, Speichern/Laden, Rendern, Kamera, Eigenschaften von Spielobjekten, Interaktion zwischen Spielobjekten, Sound, Musik, KI, Netzwerk, Pathfinding, Kollisionserkennung, Debug-Helfer, Effekte,...
 - Während der Entwicklung (**Tools**):
 - Wie bekomme ich Assets (Modelle, Texturen, Sound, etc.) ins Spiel?
 - Wie erstelle ich Assets (z.B. Level)?
 - Debug-Helfer.
- Was wird mir bereits zur Verfügung gestellt?
- Wie verbinde ich alles?

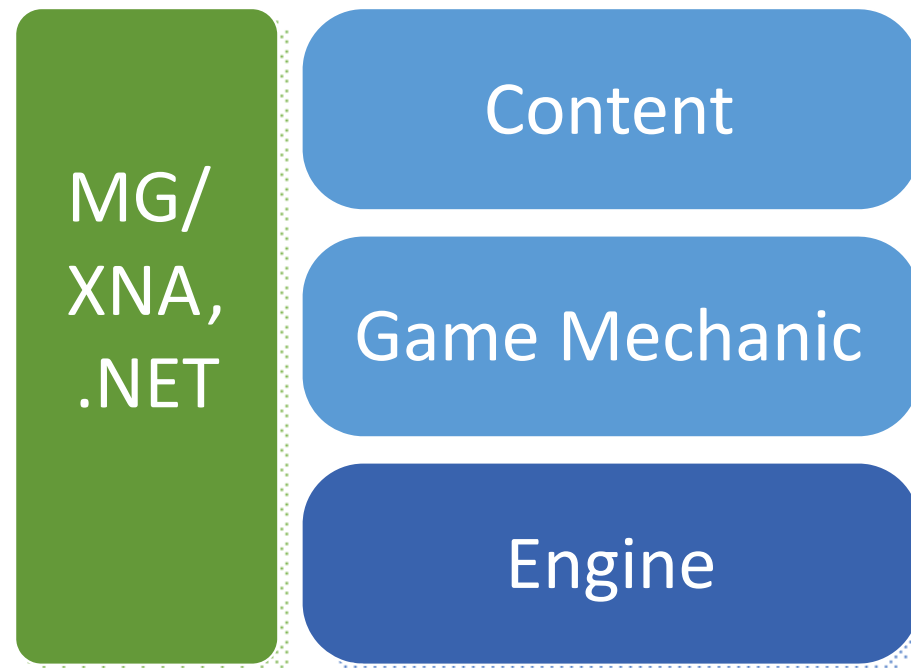


[Gregory, 2009, 29]



Wo fangen wir an?

- **Top-Down:** Zuerst Strukturen, die Dinge aufteilen.
- Ganz grobe Unterteilung:





MONOGAME/XNA UND .NET

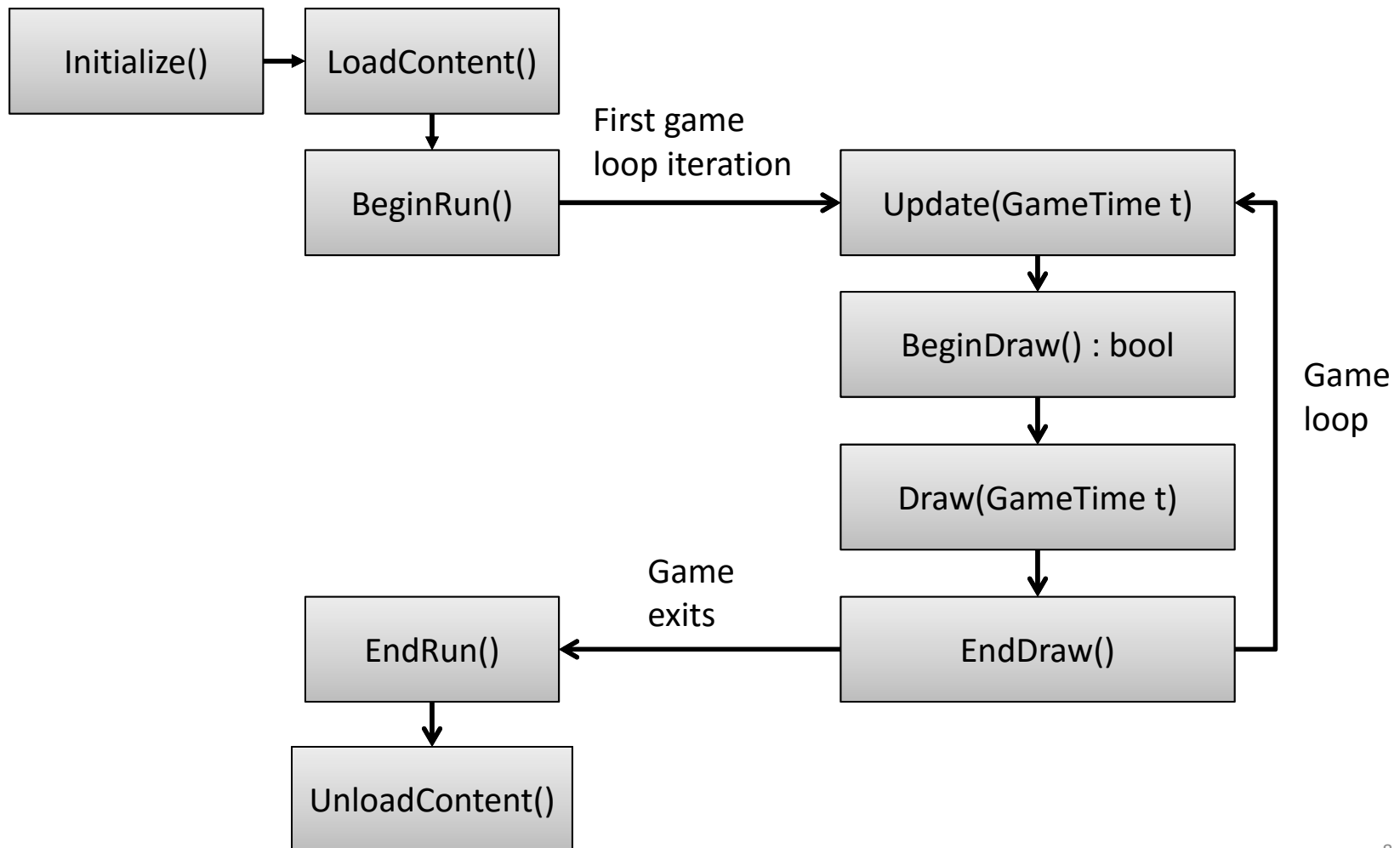


MonoGame/XNA und .NET

- **MonoGame/XNA** übernimmt bereits viele Aufgaben:
 - Abstraktion von Grafik, Sound und Input.
 - Aufbereiten von Assets (Content Pipeline).
 - Einfache Anzeigemethoden (BasicEffect, SpriteBatch).
 - Datentypen (Vector2D, 3D, Matrix, Rectangle, ...).
- **.NET** hilft ebenfalls:
 - Mathematik und Zufallszahlen.
 - Serialisierung / Deserialisierung von XML und binären Daten.
 - Diverse Datenstrukturen.
 - Debugging.



MonoGame/XNA Game Lifecycle





PUZZLESTÜCK I: SCREENS UND INPUTS

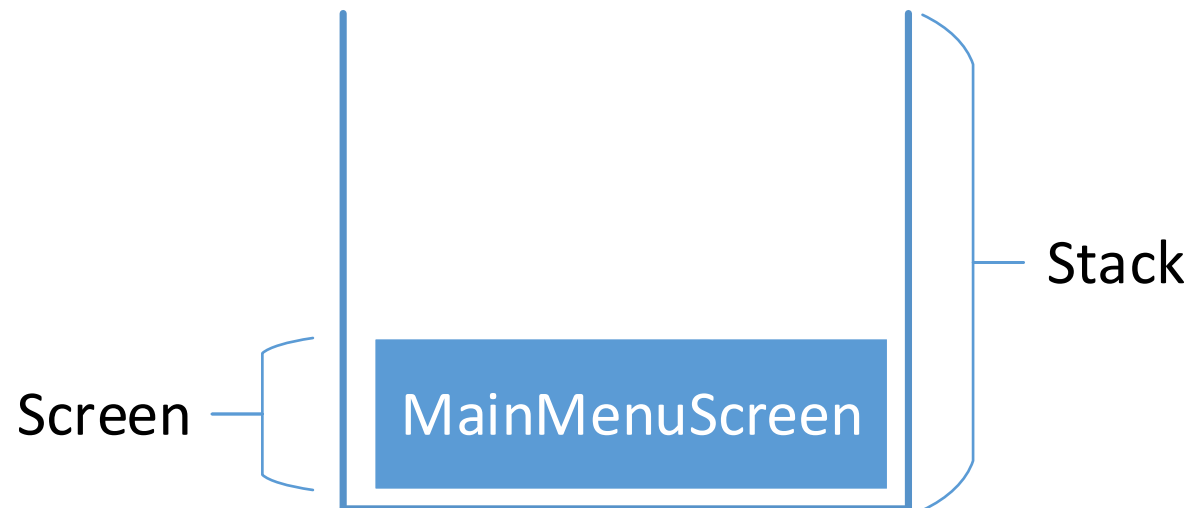


Screen Management

- Verschiedene Ansichten des Spiels in **Screens** unterteilen
 - Hauptmenü
 - Optionsmenü
 - Spielansicht
 - HUD
 - Ladescreen
 - ...
- Alle aktiven Screens werden durch einen **ScreenManager** verwaltet.
- **Übergänge zwischen Screens** durch Methoden des ScreenManagers.

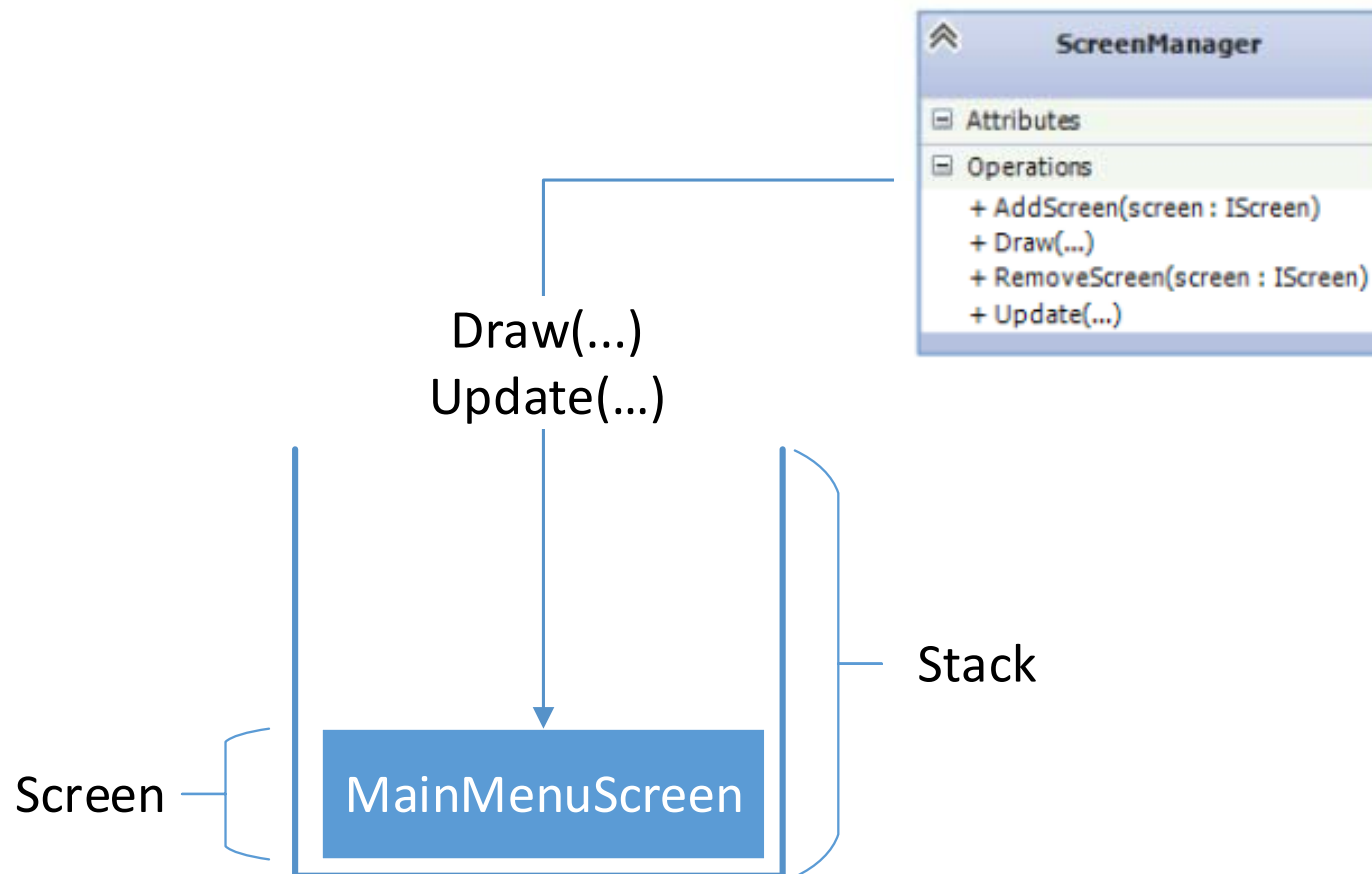


ScreenManager



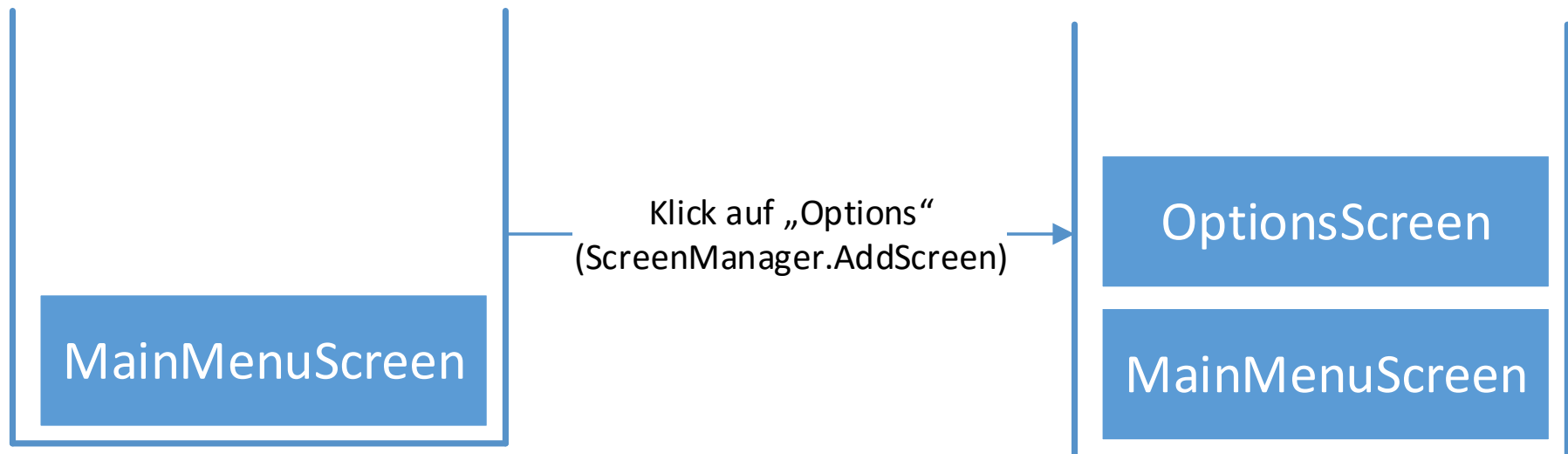


ScreenManager



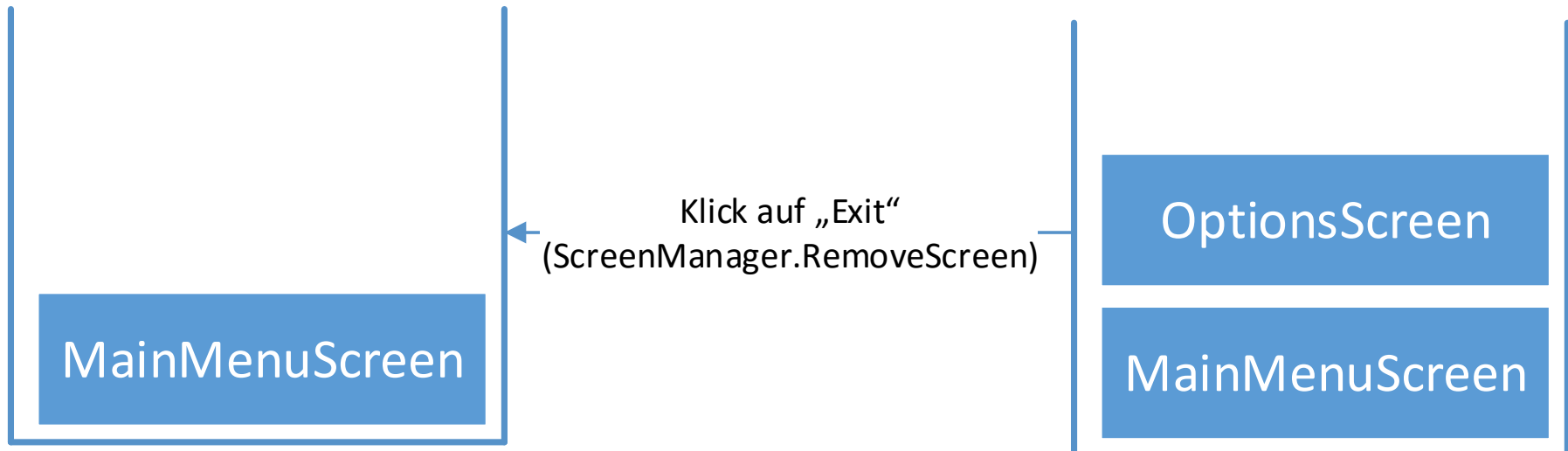


ScreenManager



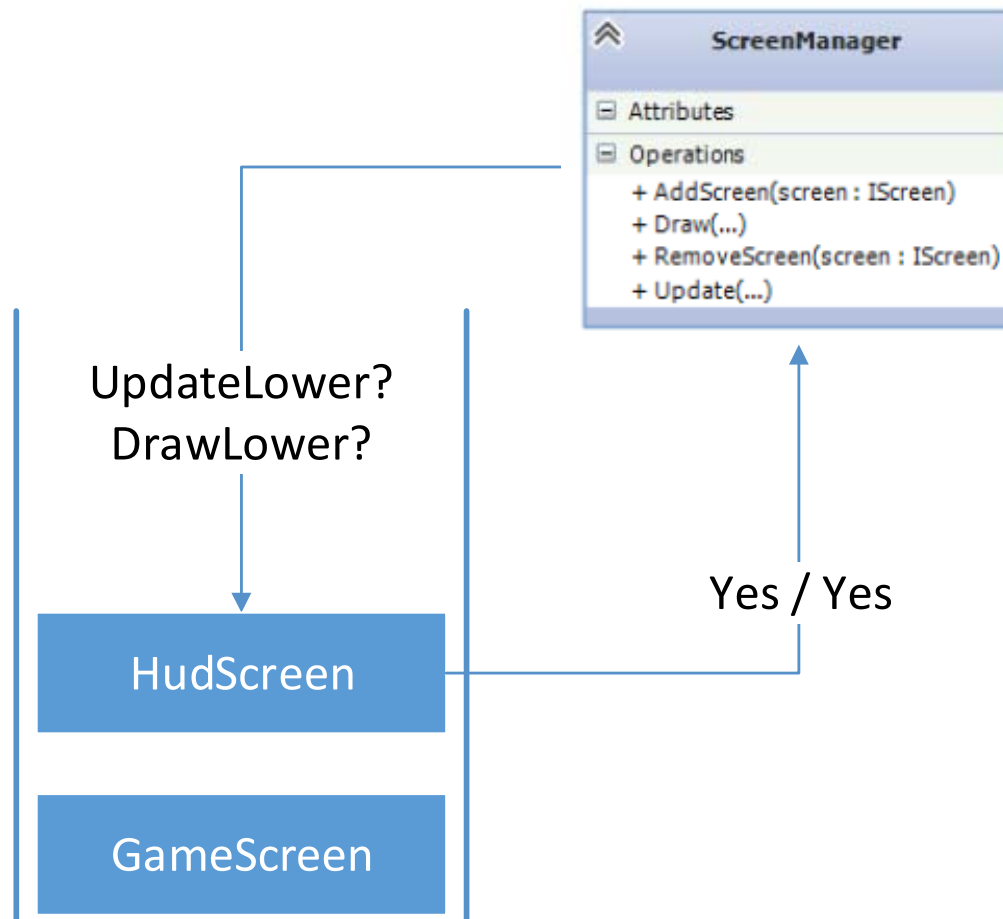


ScreenManager



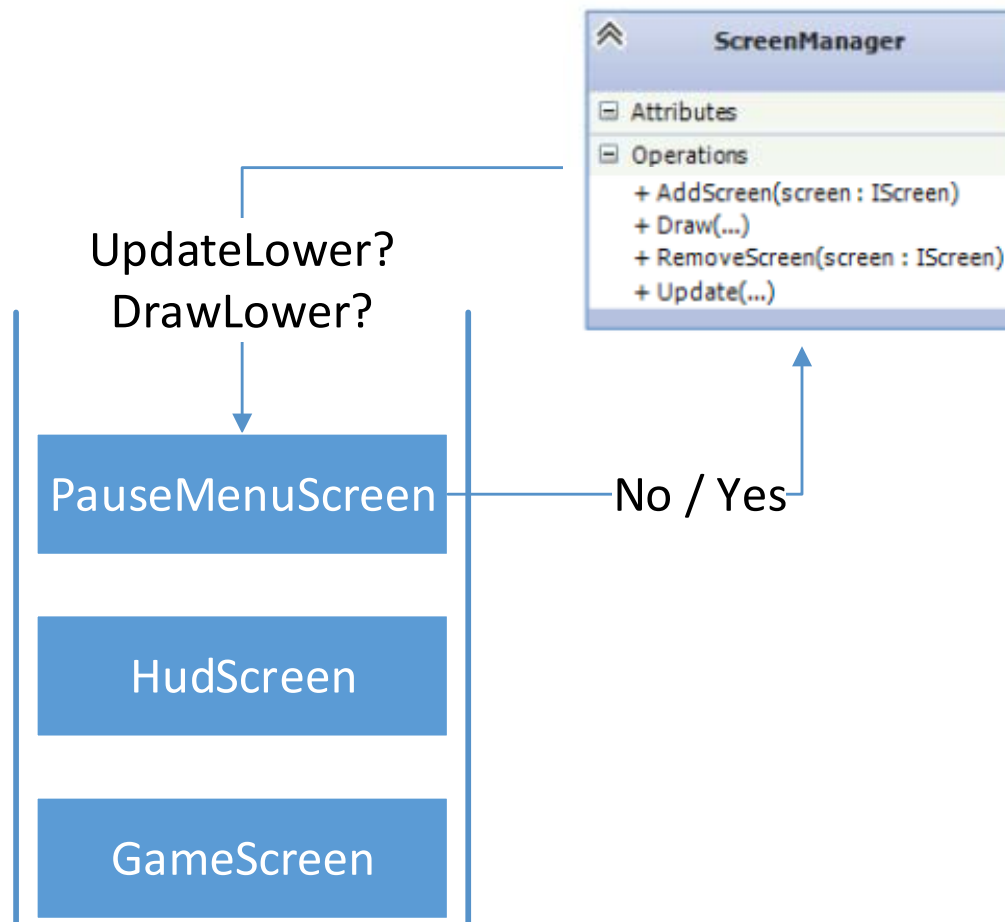


ScreenManager



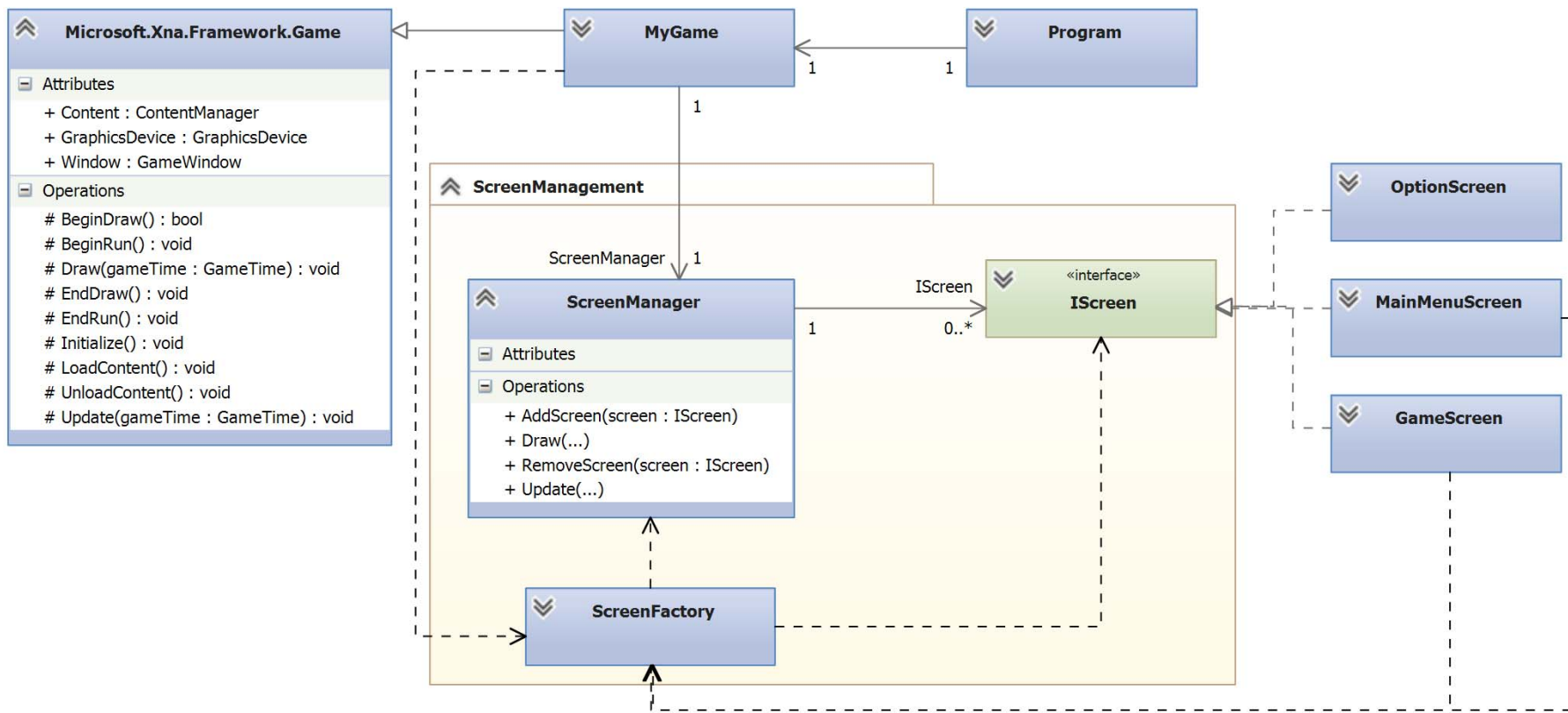


ScreenManager





ScreenManagement



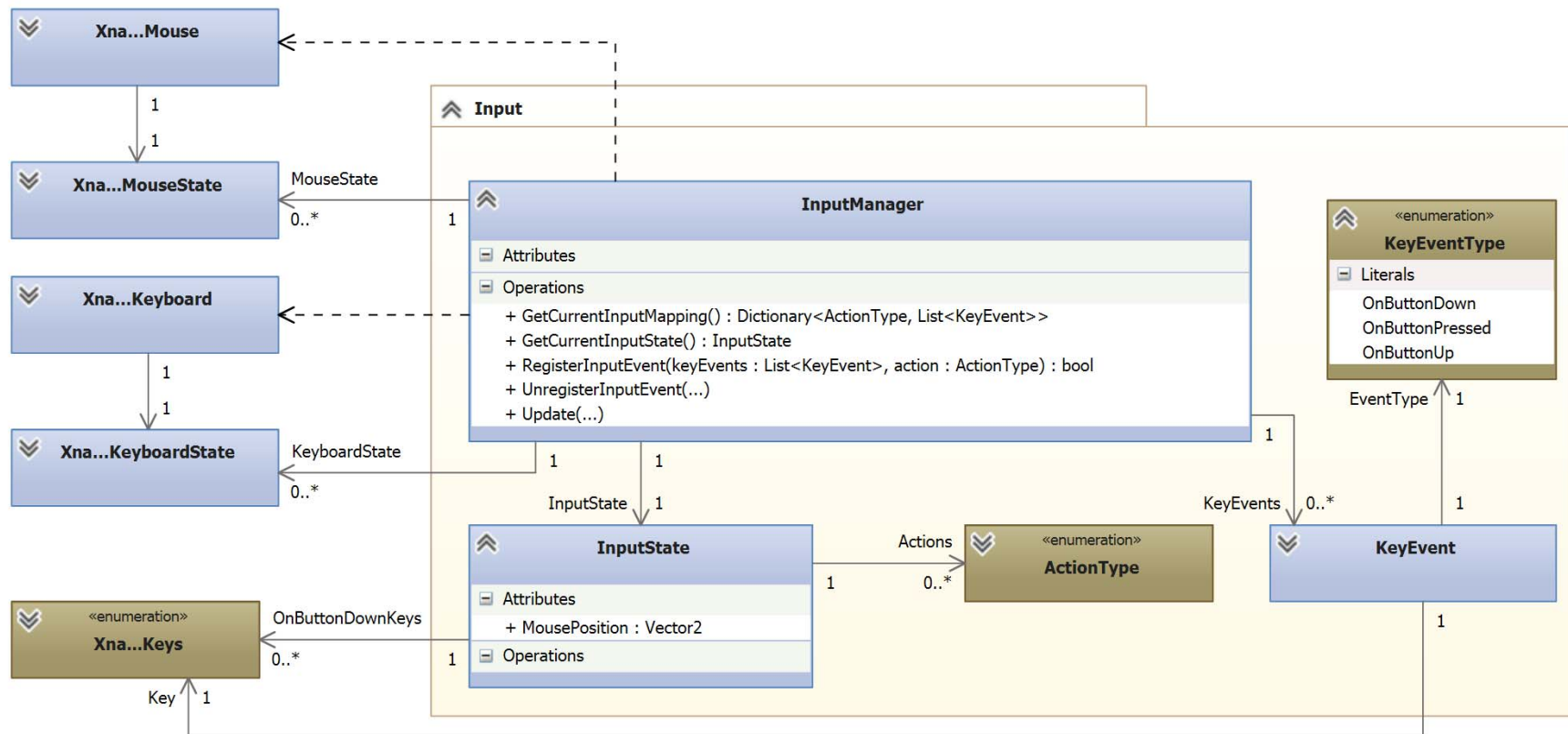


Inputs

- Keyboard, Mouse, Gamepad, etc. sind **Input Devices**.
- Ereignisse von Input Devices heißen **Inputs**.
- Inputs können in **MonoGame/XNA** durch den Microsoft.Xna.Framework.Input Namespace abgefragt werden, z.B.:
`KeyboardState k = Keyboard.GetState();`
- State enthält Informationen wie Mausposition, Zustand einer Taste (gedrückt, nicht gedrückt), etc., aber keine **Historie**.
- Wir wollen:
 - Unterscheiden zwischen Taste „gerade eben“ gedrückt, Taste losgelassen, etc.
 - Keine Inputs verpassen.
 - Aus konkreten Inputs („A“ gedrückt) abstrakte Inputs machen (z.B. AttackAction).



Inputs



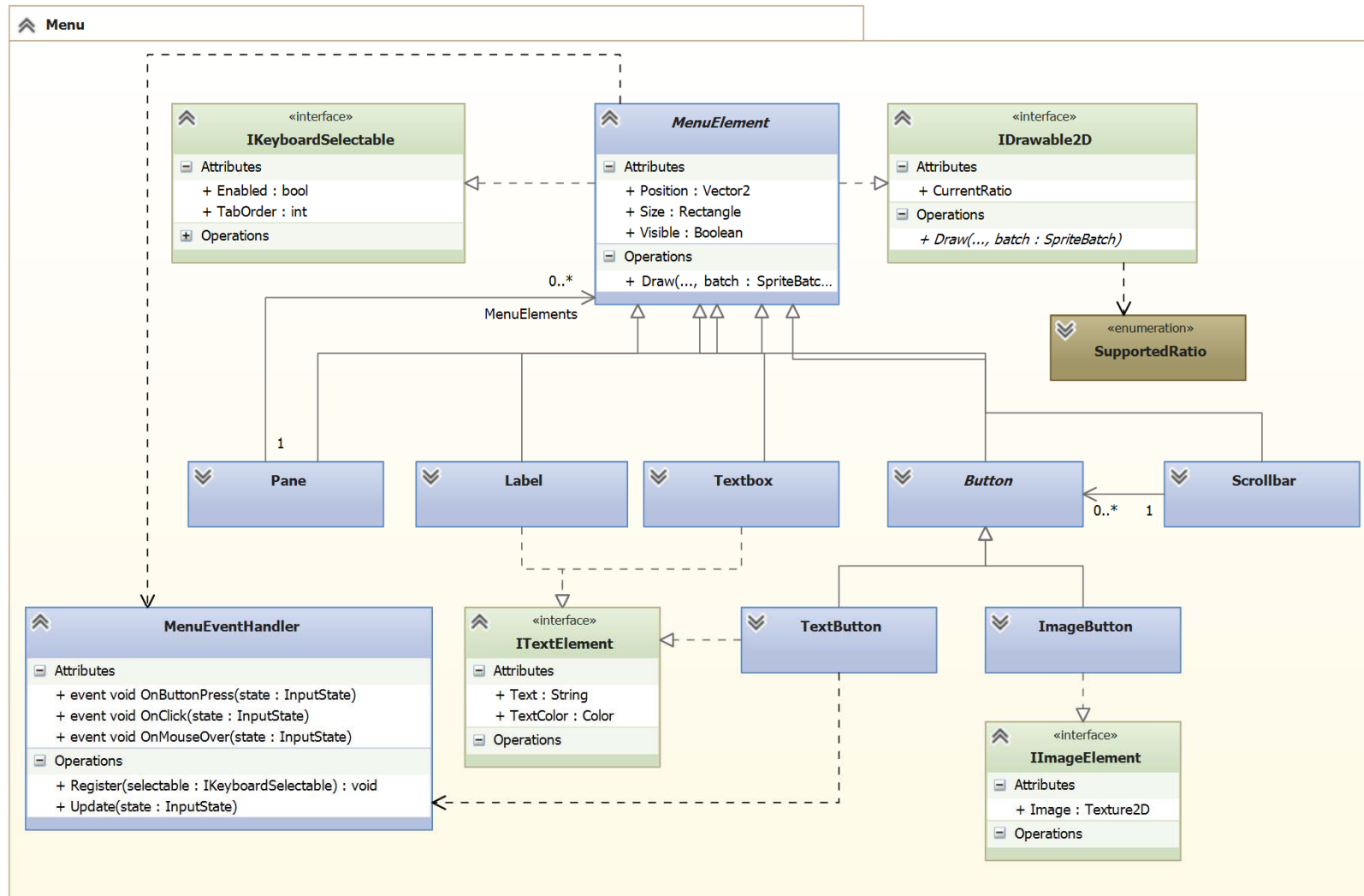


Menü

- Ein Menü besteht aus vielen **Elementen**:
Button, Label, Scrollbar, Textbox, Checkbox, ...
- Menüelemente werden nicht nur im **Hauptmenü** verwendet,
sondern auch im **HUD**.
- **Probleme**:
 - Positionierung
 - Input Handling
 - Verschiedene Auflösungen und Seitenverhältnisse



Simple Menü





Das eigentliche Spiel

- Besteht aus mehreren Screens:
 - GameScreen
 - HUD
 - Minimap
- Was muss im GameScreen alles getan werden?
 - Dinge zeichnen (Spielobjekte, Karte).
 - Aktionen des Spielers verarbeiten.
 - Spielzustand (Positionen, Ressourcen, Spielobjektzustände, etc.) verwalten.
 - ...
- Offene Probleme
 - Screens **dispatchen** nur Update und Draw. Reicht das?
 - Wie teilen mehrere Screens Informationen?
 - ...



PUZZLESTÜCK II: SPIELOBJEKTE



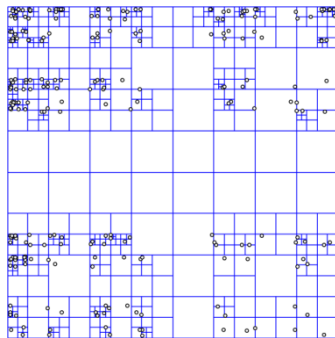
Spielobjekte

- **Spielobjekte** sind all die Dinge, die eine **Repräsentation** in der **Spielwelt** haben.
 - Charaktere, Fahrzeuge, Bäume, Raketen, Gras, Steine, Trigger, Lichtquellen, ...
- Spielobjekte müssen manchmal...
 - **gezeichnet** werden.
 - sich **bewegen**.
 - **zerstörbar** sein.
 - miteinander **kollidieren**.
 - etc.
- Wie kann ich diese vielen Objektarten und ihre Operationen **effizient** verwalten?

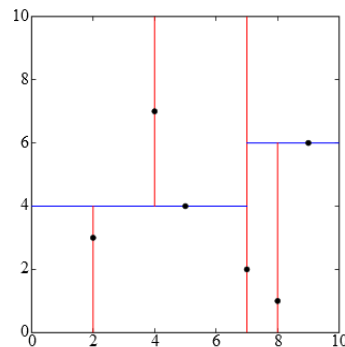


Szenengraph

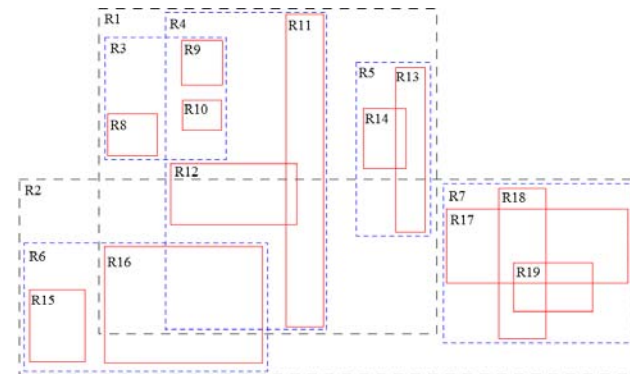
- Ein **Szenengraph** ist eine zentrale Datenstruktur, die logische und/oder räumliche Repräsentation einer Szene verwaltet.
- Wird z.B. verwendet um
 - Antworten auf **räumliche** Fragen zu beschleunigen.
 - Update- und Draw-Aufrufe an alle Spielobjekte weiterzureichen.
- **Beispiele**
Liste, Heap, Quad- / Octree, KD-Tree, R-Tree, ...



Quadtree



KD-Tree



R-Tree



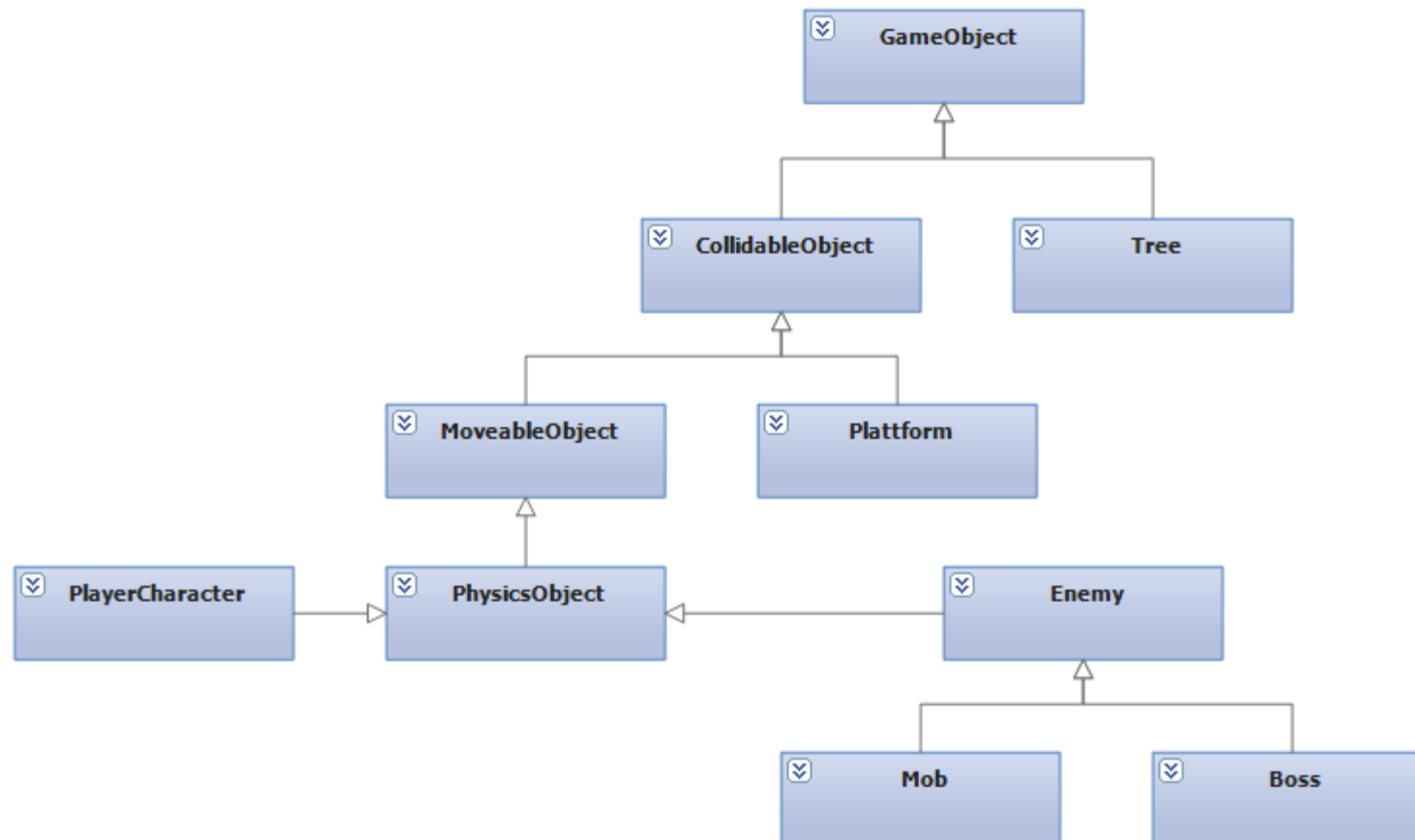
Spielobjekt-Architekturen

- **Objekt-zentriert**
 - Spielobjekte werden als Klassen implementiert und als Instanzen repräsentiert.
 - Eigenschaften und Verhalten wird durch die Klasse(n) gekapselt.
 - Spielwelt ist eine Ansammlung von Spielobjekt-Instanzen.
- **Eigenschaften-zentriert**
 - Eigenschaften der Spielobjekte werden als Tabellen gespeichert, eine pro Eigenschaft.
 - Spielobjekte sind nur eine ID.
 - Operationen sind nach wie vor Klassen/Instanzen, die diese Tabellen lesen oder modifizieren.
 - Ähnlichkeit zu relationalen Datenbanken.
- ...



Deep Hierarchy

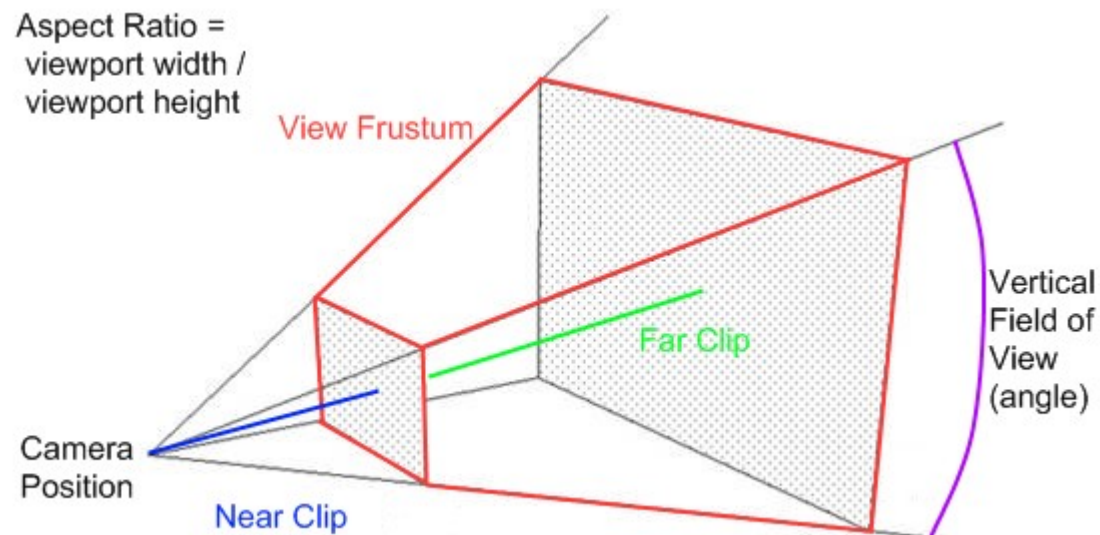
- Eine große Vererbungsstruktur für Spielobjekte.





Exkurs: Simple Kamera

- Speichert Position, ViewMatrix, ProjectionMatrix.
- Grundlage für **Picking**.
- Definiert **View Frustum**:



[<http://ksgamedev.wordpress.com/tag/maths/>]

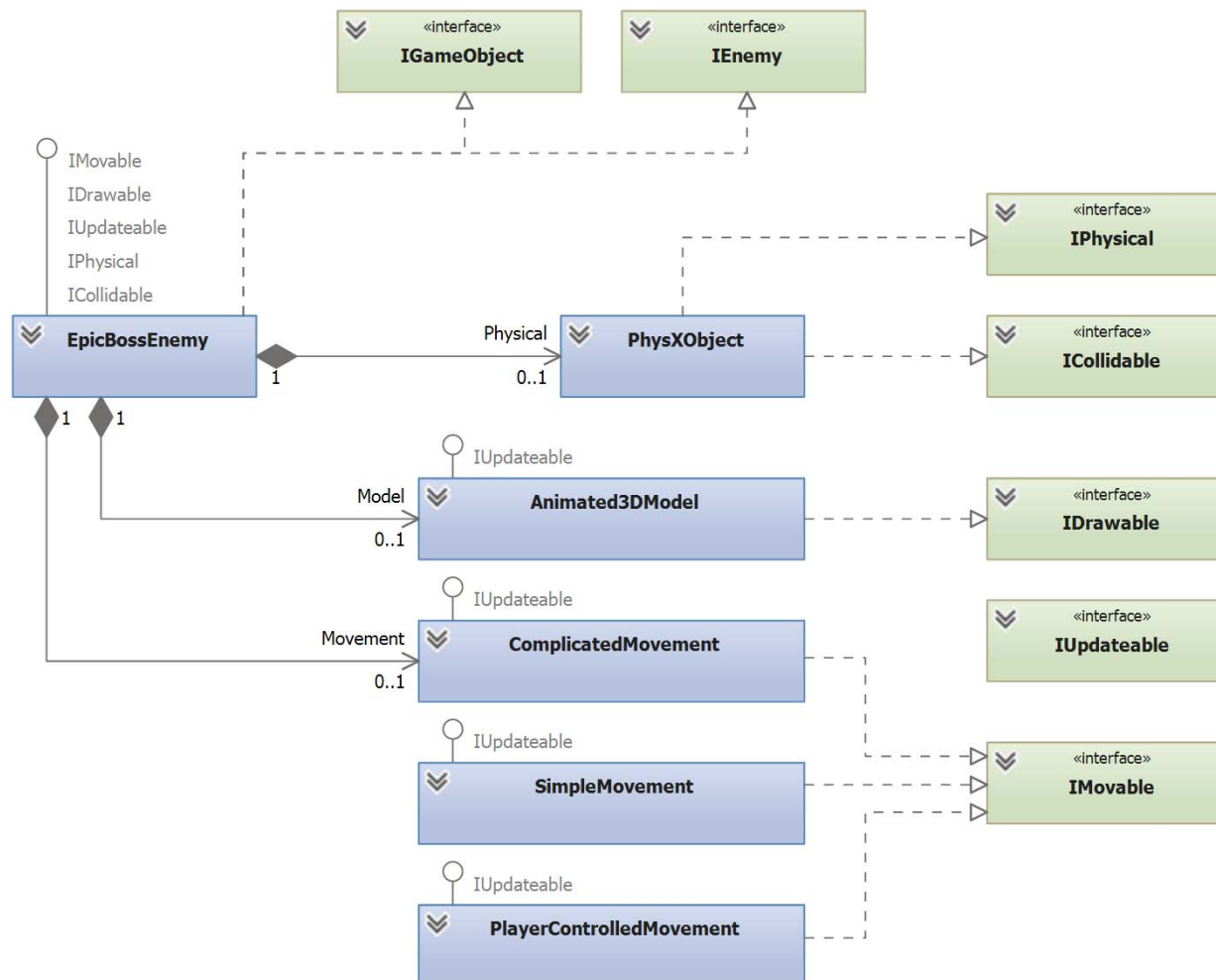


Wunsch

	Tree	Plattform	PlayerCharacter	Mob	Boss	Camera
GameObject	X	X	X	X	X	X
CollidableObject		X	X	X	X	
MoveableObject			X	X	X	X
PhysicsObject			X	X	X	
Enemy				X	X	



Spielobjekte sind Sammlungen von „Eigenschaften“ (Komposition)





PUZZLESTÜCK III: PATHFINDING

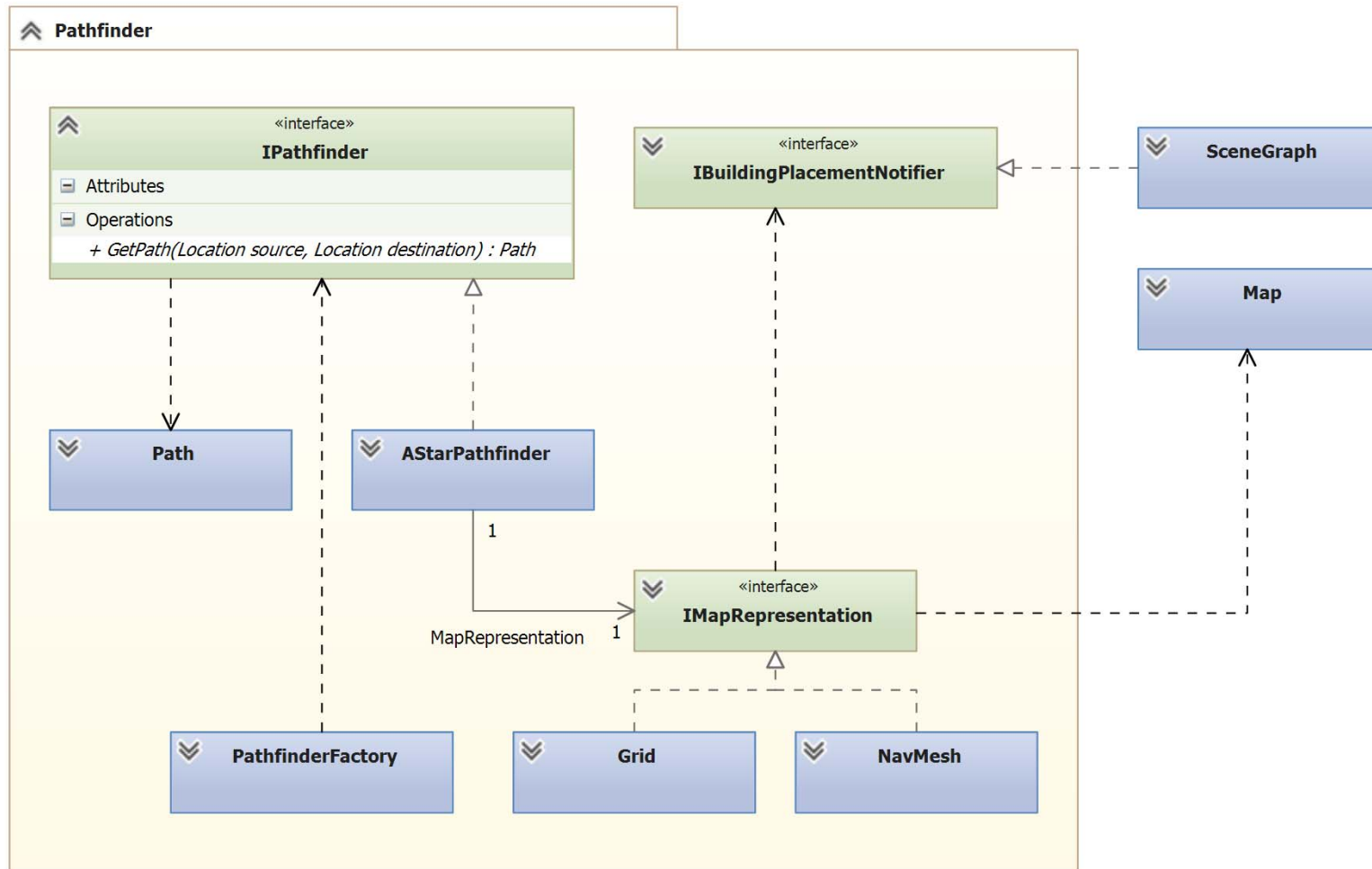


Pathfinding

- Zwei Arten: Online und Offline Pathfinding
- **Offline**
 - Berücksichtige nur die Welt und vielleicht unbewegliche Objekte.
 - Im wesentlichen immer A* als Suchalgorithmus.
 - Viele Möglichkeiten für Weltrepräsentation:
Grid, Hierarchical Grid, Waypoint Graph, Navigation Mesh, ...
- **Online**
 - Wie weiche ich beweglichen Objekten aus?
 - Viele (parametrisierbare) Möglichkeiten:
Steering, Flocking, Flow Fields, ...
 - Kann als Bewegungsmodul in Spielobjekten implementiert werden.



Pathfinding





ZUSAMMENBAU?



Ein weites Feld...

- Sehr viele **Fragen bleiben offen**.
 - Netzwerk-Multiplayer?
 - Sound?
 - Zeichnen und Grafikeffekte (Z-Buffer, Shader, Perspektiven, Partikel, Performance, ...)?
 - Online-Pathfinding?
 - KI?
 - Laden/Speichern?
 - Player?
 - Mathematik?
- Kommen Sie in die **Poolsprechstunde**, um sich speziell für Ihr Spiel beraten zu lassen.



FRAGEN?



Quellen

- [Gregory, 2009] Gregory, J. (2009). Game Engine Architecture. A K Peters Limited. ISBN 1568814135, 9781568814131.
- [<http://ksgamedev.wordpress.com/tag/maths/>]